SOUS-ROUTINES DE TRI EN MÉMOIRE CENTRALE : TRIQS et TRIPT

par

Lucie Bibeau

Université de Montréal
Centre de recherche sur les transports - Publication #309
Août 1983

Dépôt légal - 4e trimestre 1983

E R R A T U M

Veuillez remplacer, tout au long de ce document, les noms "tri de Shell de type transposition-comparaison", "tri de Shell de type T.C." et "tri de Shell" par le nom "tri par insertion simple".

De plus, la variable NSHELL utilisée dans les sous-routines TRIQS et TRIPT sera remplacée par la variable NINSER dans les fichiers correspondants.

- - - - - - - - - - - -

RÉSUMÉ

Ce document a pour but de faire connaître les sous-routines de tri TRIQS et TRIPT en décrivant, entre autres choses, la méthode de tri utilisée et l'utilisation relativement simple de ces sous-routines. Le temps de calcul requis est brièvement exposé et une extension de TRIQS et de TRIPT est discutée à la fin de cet ouvrage.

- - - - - - - - - - - - - -

TABLE DES MATIÈRES

But de TRIQS et de TRIPT 1

Contenu de TRIQS et de TRIPT 1

Méthode de tri utilisée. 2

Temps de calcul. 5

Réalisation de l'usager. 7

Extension. 10

Remerciements. 14

Annexe (Listes de TRIQS et de TRIPT) 15

Bibliographie. 24

- - - - - - - - - - - - - - -

But de TRIQS et de TRIPT :

Le but de ces sous-routines est de trier, dans un ordre croissant, un ensemble d'éléments selon certaines des informations dont ils sont composés, soit sur place (avec TRIQS), soit en ne triant que les indices de ces éléments dans un vecteur construit à cet effet (avec TRIPT).

Les deux sous-routines TRIQS et TRIPT utilisent un tableau afin d'y conserver les éléments identifiés par une ou plusieurs informations. Pour faciliter la tâche de l'usager, ces éléments à trier correspondent aux lignes du tableau et les informations associées correspondent aux colonnes. Ceci permet, pour l'usager dont un élément est défini à partir de plusieurs vecteurs, d'utiliser une suite de ces vecteurs comme tableau au lieu de redéfinir un tableau (ces vecteurs doivent tous être de même type (entier,réel, ...) pour l'utilisation de TRIQS et de TRIPT telles quelles; autrement, voir la section "Extension").

Contenu de TRIQS et de TRIPT :

TRIQS est une sous-routine qui utilise elle-même deux sous-routines et une fonction :

1) Sous-routine AFFECT servant à initialiser un tableau temporaire pour fin de comparaison.
2) Sous-routine TRANSF servant au transfert de deux éléments.
3) Fonction COMPAR servant à comparer deux éléments selon les clés établies (i.e. selon certaines des informations identifiant les éléments).

TRIPT utilise pareillement deux sous-routines et une fonction analogues à celles décrites ci-dessus (se nommant également AFFECT, TRANSF et COMPAR).

Le tout est programmé en FORTRAN IV.

Méthode de tri utilisée :

TRIQS (ainsi que TRIPT) est une combinaison des tris "quicksort" et de Shell de type transposition-comparaison.

1) Quicksort :

Les éléments sont groupés en deux sous-ensembles : un dont les éléments sont inférieurs à l'élément du milieu et l'autre dont les éléments sont supérieurs à l'élément du milieu. Le processus est recommencé pour le sous-ensemble le plus petit, pendant que les indices délimitant l'autre sous-ensemble sont gardés en mémoire dans une pile. Quand les éléments sont tous à leur place, l'indice de pile devient nul et la sous-routine renvoie les éléments triés.

2) Shell :

Ce tri est utilisé plutôt que le quicksort au cours de la procédure si le nombre d'éléments du sous-ensemble à trier est inférieur ou égal à 10 car il s'avère plus rapide pour des ensembles de cette longueur (le nombre 10 est expérimental et, de ce fait, approximatif). La méthode se résume comme suit : les éléments sont examinés un à un dans l'ensemble et si on trouve un élément inférieur à l'élément qui le précède (selon les clés utilisées), cet élément est repoussé vers le début de l'ensemble jusqu'à ce que l'élément qui le précède ne lui soit pas supérieur ou jusqu'au début de l'ensemble.

Exemple :

(tri sur place)

Soient les cinq éléments suivants, identifiés par deux informations (âge, prénom) et supposons qu'on veuille les trier selon la première information :

		éléments				
		1	2	3	4	5
informations	âge	24	16	20	5	18
	prénom	Serge	Martin	Françoise	Marie	Benoît

1° par le quicksort :

. l'élément du milieu est le $(\lfloor(1+5)/2\rfloor = 3)^{e}$ élément

. on échange le premier élément et l'élément du milieu car 24 > 20

20	16	24	5	18
Françoise	Martin	Serge	Marie	Benoît

. on échange le dernier élément et l'élément du milieu (qui est maintenant (24, Serge)) car 18 < 24

20	16	18	5	24
Françoise	Martin	Benoît	Marie	Serge

. on échange le premier élément et l'élément du milieu (qui est maintenant (18, Benoît)) car 20 > 18

18	16	20	5	24
Benoît	Martin	Françoise	Marie	Serge
↑				↑
'gauche'				'droite'

. le pointeur 'droite' (contenant l'indice de l'élément sur lequel il pointe) pointe d'abord sur le dernier élément et se déplace vers la gauche jusqu'à ce qu'il rencontre un élément inférieur ou égal à l'élément du milieu (selon l'âge)

. le pointeur 'gauche' pointe d'abord sur le premier élément et se déplace vers la droite jusqu'à ce qu'il rencontre un élément supérieur ou égal à l'élément du milieu (selon l'âge)

18	16	20	5	24
Benoît	Martin	Françoise	Marie	Serge
		↑	↑	
		'gauche'	'droite'	

. les pointeurs 'droite' et 'gauche' s'échangent leurs éléments respectifs, i.e. on échange les 3e et 4e éléments, puisque 'gauche' < 'droite'

18	16	5	20	24
Benoît	Martin	Marie	Françoise	Serge

remarque : même si,physiquement, l'élément du milieu est maintenant (5, Marie), on conserve dans le pointeur 'milieu' l'indice de l'élément du milieu précédent (i.e. (20, Françoise))

. le pointeur 'droite' est alors redéplacé vers la gauche et le pointeur 'gauche' vers la droite comme précédemment (les éléments étant comparés à (20, Françoise))

18	16	5	20	24
Benoît	Martin	Marie	Françoise	Serge
		↑	↑	
		'droite'	'gauche'	

. puisque 'gauche' = 4 > 'droite' = 3, l'ensemble des éléments est séparé en deux et le même processus est recommencé pour le plus petit sous-ensemble (i.e. les deux derniers éléments).

note : L'élément 'de droite' utilisé dans TRIQS et dans TRIPT correspond en réalité au dernier élément du tableau contenant l'ensemble des éléments à trier et celui 'de gauche' correspond au premier élément de ce tableau.

2o par Shell :

. le 1er élément n'est précédé d'aucun élément : on commence donc par le second élément et comme 16 < 24, les premier et deuxième éléments sont échangés

16	24	20	5	18
Martin	Serge	Françoise	Marie	Benoît

. comme 20 < 24 mais 20 > 16, les deuxième et troisième éléments sont échangés

16	20	24	5	18
Martin	Françoise	Serge	Marie	Benoît

. le quatrième élément est repoussé à la première position dans l'ensemble puisque 5 < 16, 20 et 24

5	16	20	24	18
Marie	Martin	Françoise	Serge	Benoît

. de même, le cinquième élément est repoussé en troisième position et les éléments sont alors triés selon l'âge

5	16	18	20	24
Marie	Martin	Benoît	Françoise	Serge

Temps de calcul :

TRIQS et TRIPT ont été testées en faisant varier le nombre d'éléments à trier, le nombre d'informations identifiant ces éléments et le nombre de clés utilisées. Les deux sous-routines ont en même temps été comparées au tri appelé sélection simple (le minimum des N éléments à trier est recherché et échangé avec le premier élément, le minimum des N-1 éléments qui restent est recherché et échangé avec le deuxième élément, etc...) fréquemment employé. Dans le cas de TRIQS et de TRIPT, comme dans

le cas de la sélection simple, le temps de calcul augmente avec le nombre d'éléments, le nombre d'informations et le nombre de clés.

Le tableau (1) montre quelques résultats, dans le cas où le tri est effectué en ne considérant qu'une seule clé. Les temps inscrits représentent une moyenne sur plusieurs valeurs obtenues (ces valeurs variant de 0% à 10% entre elles) en triant des éléments composés de nombres entiers (représentant les informations mentionnées) générés aléatoirement.

Notons que pour un nombre donné d'éléments, le temps de calcul d'un tri sur les indices ne varie que d'au plus 10%, quel que soit le nombre d'informations (puisqu'on a le même nombre de clés pour tous les tests, les comparaisons sont les mêmes et avec le tri sur indices, les échanges le sont également).

On voit d'autre part que plus le nombre d'éléments à trier est grand, plus on a avantage à utiliser le quicksort.

nombre d'éléments	tri	tri sur place — nombre d'informations					tri sur indices
		3	5	10	30	50	
100	sélection simple	0,09	0,09	0,10	0,14	0,18	0,10
	quicksort Shell	0,04	0,04	0,06	0,14	0,19	0,03
500	sélection simple	2,51	2,53	2,57	2,74	2,99	2,66
	quicksort Shell	0,29	0,35	0,47	1,01	1,45	0,25
1000	sélection simple	9,89	9,99	10,11	10,55	11,49	10,83
	quicksort Shell	0,67	0,80	1,11	2,26	3,72	0,56

Tableau (1) : Temps de calcul des tris (en secondes)

Soit N le nombre d'éléments à trier. Pour le quicksort, le nombre moyen d'opérations (comparaisons et échanges) est $O(N \log_2 N)$ et le nombre maximum d'opérations est $O(N^2)$ (ce nombre maximal est atteint si à chaque fois qu'un ensemble est divisé en deux sous-ensembles, un des sous-ensembles ne contient qu'un élément et l'autre contient le reste des éléments de l'ensemble traité). Notons que si le nombre de clés est grand par rapport à N, il faut, pour être plus précis, multiplier les nombres moyen et maximum de comparaisons déjà énoncés par ce nombre de clés.

Pour le tri de Shell, le nombre de comparaisons se situe entre (N-1) et [N(N-1)]/2 et le nombre d'échanges entre 0 et [N(N-1)]/2. Les bornes inférieures sont atteintes lorsque l'ensemble d'éléments est déjà trié et les bornes supérieures lorsque les éléments sont dans un ordre décroissant.

Pour le tri de sélection simple, le nombre de comparaisons est toujours le même, soit [N(N-1)]/2 et le nombre d'échanges se situe entre 0 et [N(N-1)]/2.

Réalisation de l'usager :

L'usager doit créer lui-même son programme principal d'où il appelle TRIQS ou TRIPT (sans paramètre) selon le cas. Dans ce programme doit se trouver le common /PARAM/ utilisé dans TRIQS et dans TRIPT. De plus, si l'usager veut l'impression de ses éléments (ou indices) ordonnés, il doit lui-même inclure les instructions nécessaires dans son programme.

1) Common /PARAM/ :

Ce common est défini dans TRIQS comme suit :

```
IMPLICIT INTEGER (A-Z)
COMMON /PARAM/  MATTRI (NELEM,NINFO), CLE (NCLE)
DIMENSION ECHAN (NINFO), TEMP (NCLE), ILOW (16), IUP (16)
DECLAR MATTRI, ECHAN, TEMP
```

et dans TRIPT comme suit :

```
IMPLICIT INTEGER (A-Z)
COMMON /PARAM/ MATTRI (NELEM,NINFO), CLE (NCLE),
+              PTEUR (NELEM)
DIMENSION TEMP (NCLE), ILOW (16), IUP (16)
DECLAR MATTRI, TEMP
```

. pour l'appel de TRIQS, l'usager doit inclure dans le common /PARAM/ du programme principal :

i) tableau dont les lignes contiennent les éléments à trier, dimensionné 'nelem' x 'ninfo'

ii) vecteur contenant les indices des informations servant de clés de longueur 'ncle' (doit être déclaré entier)

. pour l'appel de TRIPT, l'usager doit y inclure :

i) le tableau et le vecteur décrits précédemment

ii) vecteur conservant les indices des éléments de longueur 'nelem' (ce vecteur a été mis dans le common dans le but de permettre à l'usager d'imprimer ses éléments triés s'il le désire. Ce vecteur doit être déclaré entier).

où 'ninfo' est le nombre d'informations associées à chaque élément (doit être le même pour tous les éléments)

'ncle' est le nombre de clés

'nelem' est le nombre d'éléments à trier

Remarques :

i) 'ncle' doit être inférieur ou égal à 'ninfo'

ii) Les tableaux TEMP, ILOW, IUP et ECHAN mentionnés précédemment sont décrits dans les sous-routines TRIQS et TRIPT. L'usager n'a pas à s'occuper explicitement de ces tableaux, sauf si son nombre d'éléments à trier est supérieur à $(2^{17}-1)$ (il doit alors augmenter les dimensions de ILOW et de IUP poun être certain de ne pas avoir de débordement de tableaux).

2) L'usager doit modifier dans tout TRIQS (ou TRIPT) :

nelem par le nombre d'éléments à trier

ninfo par le nombre d'informations associées à chaque élément

ncle par le nombre de clés que l'usager veut utiliser pour effectuer le tri

declar par real ou integer (ou autre chose) dépendant si les éléments à trier sont réels ou entiers.

3) Initialisation :

L'usager doit initialiser, dans son programme principal, le tableau d'éléments à trier et le vecteur contenant les indices des informations servant de clés, que ce soit TRIQS ou TRIPT qu'il utilise.

4) Localisation des fichiers :

TRIQS et TRIPT se trouvent dans des fichiers de mêmes noms sur le compte de l'usager 2403.

L'usager doit ramener le fichier désiré sur son compte pour pouvoir effectuer les modifications nécessaires :

```
      OU TRIQS U = 2403•FE
ou    OU TRIPT U = 2403•FE
```

Et l'usager peut ensuite inclure ce nouveau fichier après son programme principal, en faisant par exemple :

```
          :
programme principal
          :
*incl 1 * n = TRIQS (ou TRIPT)
```

Extension :

Si l'usager veut trier des éléments composés d'informations de différents types, il doit effectuer quelques changements dans la sous-routine de tri employée (voir listes de TRIQS et de TRIPT en annexe) :

1. S'il s'agit de TRIQS, l'usager doit :
   - redéfinir certains tableaux (MATTRI, TEMP, ECHAN) et redéfinir ou éliminer le tableau CLE
   - modifier les sous-routines TRANSF et AFFECT et la fonction COMPAR
   - modifier les paramètres de AFFECT et de COMPAR
   - modifier certaines instructions concernant le tri de Shell.
2. S'il s'agit de TRIPT, l'usager doit :
   - redéfinir certains tableaux (MATTRI, TEMP) et redéfinir ou éliminer le tableau CLE
   - modifier la sous-routine AFFECT et la fonction COMPAR
   - modifier les paramètres de AFFECT et COMPAR.

Par exemple, supposons qu'à chaque élément à trier correspondent trois informations dont la seconde est représentée par un nombre réel et

les deux autres par des nombres entiers. Un élément de ce genre serait (29, 7.3, 14). Supposons de plus qu'on veuille trier cent éléments selon les deux premières informations:

Utilisant TRIQS, on pourrait alors (entre autres choses) :

i) remplacer dans les déclarations et/ou le common :

. le tableau MATTRI contenant les éléments à trier par autant de vecteurs qu'il y a d'informations et déclarer ces vecteurs séparément comme il se doit

. le tableau TEMP servant à conserver les clés d'un élément qu'on veut comparer aux autres par autant de variables qu'il y a de clés et déclarer chacune de ces variables comme il se doit

. remplacer le tableau ECHAN servant au transfert de deux éléments par autant de variables qu'il y a d'informations et déclarer ces variables comme il se doit

ii) éliminer le tableau CLE

iii) modifier TRANSF, AFFECT et COMPAR de façon à utiliser les nouveaux vecteurs et les nouvelles variables

iv) remplacer le paramètre TEMP par les variables correspondantes dans les appels et les définitions de AFFECT et de COMPAR

v) modifier les échanges du tri de Shell de façon à utiliser les vecteurs et les variables décrits en i)

Ceci donnerait, dans le cas présent, les modifications suivantes :

. déclarations de la sous-routine TRIQS :

```
IMPLICIT INTEGER (A-Z)
COMMON /PARAM/ VECT1(00100),VECT2(00100),VECT3(00100)
DIMENSION ILOW(16),IUP(16)
REAL VECT2,TEMP2,ECHAN2
DATA NSHELL /10/
```

. les sous-routines TRANSF et AFFECT et la fonction COMPAR suivantes :

```
SUBROUTINE TRANSF(ITRAN1,ITRAN2)

IMPLICIT INTEGER (A-Z)
COMMON /PARAM/ VECT1(00100),VECT2(00100),VECT3(00100)
REAL VECT2,BIDON2

BIDON = VECT1(ITRAN1)
VECT1(ITRAN1) = VECT1(ITRAN2)
VECT1(ITRAN2) = BIDON
BIDON2 = VECT2(ITRAN1)
VECT2(ITRAN1) = VECT2(ITRAN2)
VECT2(ITRAN2) = BIDON2
BIDON = VECT3(ITRAN1)
VECT3(ITRAN1) = VECT3(ITRAN2)
VECT3(ITRAN2) = BIDON

RETURN
END

SUBROUTINE AFFECT(IAFFEC,TEMP1,TEMP2)

IMPLICIT INTEGER (A-Z)
COMMON /PARAM/ VECT1(00100),VECT2(00100),VECT3(00100)
REAL VECT2,TEMP2

TEMP1 = VECT1(IAFFEC)
TEMP2 = VECT2(IAFFEC)

RETURN
END
```

```
      INTEGER FUNCTION COMPAR(ICOMP,TEMP1,TEMP2)
      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/VECT1(00100),VECT2(00100),VECT3(00100)
      REAL VECT2,TEMP2
      IF(VECT1(ICOMP)-TEMP1) 20,10,30
   10 IF(VECT2(ICOMP)-TEMP2) 20,15,30
   15 COMPAR = 0
      RETURN
   20 COMPAR = -1
      RETURN
   30 COMPAR = 1
      RETURN
      END
```

. les appels de AFFECT et de COMPAR seraient de la forme :

```
      CALL AFFECT(MILIEU,TEMP1,TEMP2)
      IF(COMPAR(BINF,TEMP1,TEMP2)) 20,20,15
```

. et les échanges du tri de Shell pourraient se faire comme suit:

```
   95 GAUCHE = BINF
      ECHAN1 = VECT1(GAUCHE+1)
      ECHAN2 = VECT2(GAUCHE+1)
      ECHAN3 = VECT3(GAUCHE+1)
  100 VECT1(GAUCHE+1) = VECT1(GAUCHE)
      VECT2(GAUCHE+1) = VECT2(GAUCHE)
      VECT3(GAUCHE+1) = VECT3(GAUCHE)
      GAUCHE = GAUCHE - 1
      IF(GAUCHE.EQ.0) GOTO 110
      IF(COMPAR(GAUCHE,TEMP1,TEMP2)) 110,110,100
  110 VECT1(GAUCHE+1) = ECHAN1
      VECT2(GAUCHE+1) = ECHAN2
      VECT3(GAUCHE+1) = ECHAN3
```

Note : Si TRIPT était utilisé, les échanges (tri de Shell, TRANSF) n'auraient pas à être modifiés car ils sont effectués sur les indices des éléments dans cette sous-routine.

REMERCIEMENTS

Je tiens à remercier M. Réjean Lessard pour l'aide qu'il a apportée à l'élaboration des sous-routines de tri présentées dans ce document.

- - - - - - - - - - - - - -

ANNEXE

Listes de TRIQS et de TRIPT

```
      SUBROUTINE TRIGS
*
*     CETTE SOUS-ROUTINE TRIE LES ELEMENTS SELON LES CLES DESI-
*     REES EN UTILISANT LA METHODE DE TRI QUICKSORT :
*        LES ELEMENTS SONT GROUPES EN DEUX SOUS-ENSEMBLES, UN
*        DONT LES ELEMENTS SONT INFERIEURS A L'ELEMENT DU MILIEU
*        ET L'AUTRE DONT LES ELEMENTS SONT SUPERIEURS A L'ELEMENT
*        DU MILIEU. LE PROCESSUS EST RECOMMENCE POUR LE SOUS-EN-
*        SEMBLE LE PLUS PETIT, PENDANT QUE LES INDICES DELIMITANT
*        L'AUTRE SOUS-ENSEMBLE SONT GARDES EN MEMOIRE DANS UNE
*        PILE. QUAND LES ELEMENTS SONT TOUS A LEUR PLACE, L'INDICE
*        DE PILE DEVIENT NUL ET LA SOUS-ROUTINE RENVOIE LES ELE-
*        MENTS TRIES.
*     SI LE NOMBRE D'ELEMENTS DU SOUS-ENSEMBLE A TRIER <= 10 (NSHELL),
*     UN TRI DE SHELL DE TYPE T.C. (TRANSPOSITION-COMPARAISON) EST U-.
*     TILISE.
*     LORSQUE SEUL LE TRI QUICKSORT EST UTILISE, DES PILES DE DI-
*     MENSION 'K' PERMETTENT DE TRIER JUSQU'A [2**(K+1) - 1] ELE-
*     MENTS. ( NOTE : PUISQU'ICI LE TRI DE SHELL EST EMPLOYE POUR
*     DES PETITS SOUS-ENSEMBLES, ON PEUT GENERALEMENT, EN FAISANT
*     ATTENTION AUX DEBORDEMENTS DE TABLEAUX, TRIER PLUS DE
*     [2**(K+1) - 1] ELEMENTS AVEC DES PILES DE CETTE LONGUEUR).

      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM,NINFO),CLE(NCLE)
      DIMENSION ECHAN(NINFO),TEMP(NCLE),ILOW(16),IUP(16)
      DECLAR MATTRI,ECHAN,TEMP
      DATA NSHELL /10/

****      VARIABLES DU COMMON      ****

*     CLE : TABLEAU DONT LE IEME INDICE INDIQUE QUEL VECTEUR-CO-
*           LONNE DE 'MATTRI' SERT DE IEME CLE POUR LE TRI
*     MATTRI : MATRICE CONTENANT LES ELEMENTS (VECTEURS-LIGNES)
*              A TRIER SELON CERTAINES DES INFORMATIONS (VECTEURS-
*              COLONNES)

****      VARIABLES LOCALES      ****

*     BINF : BORNE INFERIEURE DES INDICES DES ELEMENTS DU SOUS-
*            ENSEMBLE A TRIER
*     BSUP : BORNE SUPERIEURE DES INDICES DES ELEMENTS DU SOUS-
*            ENSEMBLE A TRIER
*     DROITE : INDICE DE L'ELEMENT DE DROITE QU'ON COMPARE AVEC
*              L'ELEMENT DU MILIEU, DANS LE SOUS-ENSEMBLE A TRIER;
*              VAUT 'BSUP' AU DEBUT DU TRI ET PEUT DIMINUER PAR LA
*              SUITE
*     ECHAN : TABLEAU TEMPORAIRE SERVANT AU TRANSFERT DE DEUX
*             ELEMENTS
*     GAUCHE : INDICE DE L'ELEMENT DE GAUCHE QU'ON COMPARE AVEC
*              L'ELEMENT DU MILIEU, DANS LE SOUS-ENSEMBLE A TRIER;
*              VAUT 'BINF' AU DEBUT DU TRI ET PEUT AUGMENTER PAR
*              LA SUITE
*     ILOW : VECTEUR SERVANT DE PILE POUR EMMAGASINER L'INDICE DE
*            DEBUT DU SOUS-ENSEMBLE EN ATTENTE D'ETRE TRIE
*     IPILE : INDICE DES VECTEURS SERVANT DE PILE
*     IUP : VECTEUR SERVANT DE PILE POUR EMMAGASINER L'INDICE DE
*           FIN DU SOUS-ENSEMBLE EN ATTENTE D'ETRE TRIE
*     MILIEU : INDICE DE L'ELEMENT DU MILIEU DU SOUS-ENSEMBLE A
*              TRIER
*     NSHELL : NOMBRE MAXIMUM D'ELEMENTS PERMIS POUR L'UTILISATION
*              DU TRI DE SHELL
*     TEMP : TABLEAU TEMPORAIRE UTILISE POUR LA COMPARAISON DES
*            ELEMENTS SELON LES DIFFERENTES CLES

      IPILE = 1
      BINF = 1
      BSUP = NELEM

      IF(BINF.GE.BSUP) GOTO 70
```

```
*    (SUITE DE LA SOUS-ROUTINE TRIQS)

*    COMPARAISON DE L'ELEMENT INDICE 'BINF' ET DE CELUI INDICE
*    'MILIEU' ET TRANSFERT SI NECESSAIRE

   10 CAUCHE = BINF
      MILIEU = (BINF+BSUP)/2
      CALL AFFECT(MILIEU,TEMP)
      IF(COMPAR(BINF,TEMP)) 20,20,15
   15 CALL TRANSF(BINF,MILIEU)

*    COMPARAISON DE L'ELEMENT INDICE 'BSUP' ET DE CELUI INDICE
*    'MILIEU' ET TRANSFERT SI NECESSAIRE

      CALL AFFECT(MILIEU,TEMP)
   20 DROITE = BSUP
      IF(COMPAR(BSUP,TEMP)) 25,40,40
   25 CALL TRANSF(BSUP,MILIEU)

*    SECONDE COMPARAISON DE L'ELEMENT INDICE 'BINF' ET DE CELUI
*    INDICE 'MILIEU' (CAR CE DERNIER A CHANGE) ET TRANSFERT SI
*    NECESSAIRE

      CALL AFFECT(MILIEU,TEMP)
      IF(COMPAR(BINF,TEMP)) 40,40,30
   30 CALL TRANSF(BINF,MILIEU)
      CALL AFFECT(MILIEU,TEMP)
      GOTO 40

   35 CALL TRANSF(DROITE,GAUCHE)

*    TANT QUE L'ELEMENT INDICE 'DROITE' > CELUI INDICE 'MILIEU'
*    (SELON LES CLES DEFINIES), DIMINUTION DE 'DROITE' ET COM-
*    PARAISON DE CES ELEMENTS

   40 DROITE = DROITE - 1
      IF(COMPAR(DROITE,TEMP)) 50,50,40

*    TANT QUE L'ELEMENT INDICE 'GAUCHE' < CELUI INDICE 'MILIEU'
*    (SELON LES CLES DEFINIES), AUGMENTATION DE 'GAUCHE' ET
*    COMPARAISON DE CES ELEMENTS

   50 CAUCHE = GAUCHE + 1
      IF(COMPAR(GAUCHE,TEMP)) 50,55,55

*    COMPARAISON DES INDICES 'DROITE' ET 'GAUCHE' : LORSQUE
*    CEUX-CI SE RENCONTRENT, LES ELEMENTS ONT TOUS ETE EXAMI-
*    NES ET ON SEPARE LE SOUS-ENSEMBLE EN DEUX

   55 IF(GAUCHE.LE.DROITE) GOTO 35
      IF(DROITE-BINF.LE.BSUP-GAUCHE) GOTO 60
      ILOW(IPILE) = BINF
      IUP(IPILE) = DROITE
      BINF = GAUCHE
      IPILE = IPILE + 1
      GOTO 80

   60 ILOW(IPILE) = GAUCHE
      IUP(IPILE) = BSUP
      BSUP = DROITE
      IPILE = IPILE + 1
      GOTO 80

*    FIN DU TRI D'UN SOUS-ENSEMBLE : ON DEPILE POUR VOIR S'IL
*    RESTE D'AUTRES SOUS-ENSEMBLES A TRIER

   70 IPILE = IPILE - 1
      IF(IPILE.EQ.0) RETURN
      BINF = ILOW(IPILE)
      BSUP = IUP(IPILE)
```

```
*    (SUITE DE LA SOUS-ROUTINE TRIQS)

*     TRI DU SOUS-ENSEMBLE LE PLUS PETIT FORME DE LA DIVISION DE
*     L'ANCIEN SOUS-ENSEMBLE EN DEUX; TRI DE SHELL DE TYPE T.C.
*     SI LE NOMBRE D'ELEMENTS DU SOUS-ENSEMBLE <= 10

   80 IF(BSUP-BINF.GE.NSHELL) GOTO 10
      BINF = BINF-1

   90 BINF = BINF + 1
      IF(BINF.EQ.BSUP) GOTO 70
      CALL AFFECT(BINF+1,TEMP)
      IF(COMPAR(BINF,TEMP)) 90,90,95

   95 GAUCHE = BINF
      DO 96 I=1,NINFO
      ECHAN(I) = MATTRI(GAUCHE+1,I)
   96 CONTINUE

  100 DO 106 I=1,NINFO
      MATTRI(GAUCHE+1,I) = MATTRI(GAUCHE,I)
  106 CONTINUE

      GAUCHE = GAUCHE - 1
      IF(GAUCHE.EQ.0) GOTO 110
      IF(COMPAR(GAUCHE,TEMP)) 110,110,100

  110 DO 116 I=1,NINFO
      MATTRI(GAUCHE+1,I) = ECHAN(I)
  116 CONTINUE

      GOTO 90

      END


      SUBROUTINE AFFECT(IAFFEC,TEMP)

*     CETTE SOUS-ROUTINE AFFECTE LES INFORMATIONS SERVANT DE
*    CLES, POUR L'ELEMENT DE LA MATRICE A TRIER DONT L'INDICE
*    EST 'IAFFEC', AU TABLEAU TEMPORAIRE 'TEMP'

      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM,NINFO),CLE(NCLE)
      DIMENSION TEMP(NCLE)
      DECLAR MATTRI,TEMP

****     VARIABLES LOCALES     ****

*     IAFFEC :  INDICE DE L'ELEMENT QU'ON VEUT COMPARER AVEC UN
*               AUTRE ET DONT ON AFFECTE LES INFORMATIONS LE DE-
*               FINISSANT ET SERVANT DE CLES AU TABLEAU 'TEMP'
*     KLE : VARIABLE TEMPORAIRE CONTENANT LA IEME CLE

      DO 10 I=1,NCLE
      KLE = CLE(I)
      TEMP(I) = MATTRI(IAFFEC,KLE)
   10 CONTINUE

      RETURN
      END
```

```
      INTEGER FUNCTION COMPAR(ICOMP, TEMP)
*     CETTE FONCTION COMPARE DEUX ELEMENTS SELON LES CLES DEJA
*    DEFINIES
      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM, NINFO), CLE(NCLE)
      DIMENSION TEMP(NCLE)
      DECLAR MATTRI, TEMP
****     VARIABLES LOCALES     ****
*     ICOMP : INDICE DE L'ELEMENT COMPARE, SELON CHAQUE CLE, AU
*             TABLEAU TEMPORAIRE 'TEMP'
*     KLE : VARIABLE TEMPORAIRE CONTENANT LA IEME CLE
      DO 10 I=1, NCLE
      KLE = CLE(I)
      IF(MATTRI(ICOMP, KLE)-TEMP(I)) 20, 10, 30
   10 CONTINUE
      COMPAR = 0
      RETURN
   20 COMPAR = -1
      RETURN
   30 COMPAR = 1
      RETURN
      END

      SUBROUTINE TRANSF(ITRAN1, ITRAN2)
*     CETTE SOUS-ROUTINE FAIT LE TRANSFERT DES DEUX ELEMENTS
*    DONT LES INDICES SONT 'ITRAN1' ET 'ITRAN2'
      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM, NINFO), CLE(NCLE)
      DECLAR MATTRI, BIDON
****     VARIABLES LOCALES     ****
*     BIDON : VARIABLE TEMPORAIRE SERVANT AU TRANSFERT
*     ITRAN1, ITRAN2 :  INDICES DES ELEMENTS POUR LESQUELS ON VEUT
*                       EFFECTUER UN TRANSFERT
      DO 10 I=1, NINFO
      BIDON = MATTRI(ITRAN1, I)
      MATTRI(ITRAN1, I) = MATTRI(ITRAN2, I)
      MATTRI(ITRAN2, I) = BIDON
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRIPT
*     CETTE SOUS-ROUTINE TRIE LES ELEMENTS SELON LES CLES DESIREES
*     PAR LA METHODE DE TRI QUICKSORT, EN NE DEPLACANT QUE LES POIN-
*     TEURS ASSOCIES A CES ELEMENTS :
*        LES POINTEURS SONT GROUPES EN DEUX SOUS-ENSEMBLES, UN
*        DONT LES ELEMENTS CORRESPONDANTS SONT INFERIEURS A L'ELE-
*        MENT ASSOCIE AU POINTEUR DU MILIEU ET L'AUTRE DONT LES ELE-
*        MENTS CORRESPONDANTS SONT SUPERIEURS A L'ELEMENT ASSOCIE
*        AU POINTEUR DU MILIEU. LE PROCESSUS EST RECOMMENCE POUR LE
*        SOUS-ENSEMBLE LE PLUS PETIT, PENDANT QUE LES INDICES DELI-
*        MITANT L'AUTRE SOUS-ENSEMBLE SONT GARDES EN MEMOIRE DANS
*        UNE PILE. QUAND LES POINTEURS DES ELEMENTS SONT TOUS A LEUR
*        PLACE, L'INDICE DE PILE DEVIENT NUL ET LA SOUS-ROUTINE REN-
*        VOIE LES POINTEURS TRIES (I.E. 'PTEUR(I)' CORRESPOND AU IEME
*        ELEMENT SELON L'ORDRE DEMANDE)
*     SI LE NOMBRE D'ELEMENTS (OU DE POINTEURS) DU SOUS-ENSEMBLE A
*     TRIER <= 10 (NSHELL), UN TRI DE SHELL DE TYPE T.C. (TRANSPOSI-
*     TION-COMPARAISON) EST UTILISE.
*     LORSQUE SEUL LE TRI QUICKSORT EST UTILISE, DES PILES DE DI-
*     MENSION 'K' PERMETTENT DE TRIER JUSQU'A [2**(K+1) - 1] ELE-
*     MENTS. ( NOTE : PUISQU'ICI LE TRI DE SHELL EST EMPLOYE POUR
*     DES PETITS SOUS-ENSEMBLES, ON PEUT GENERALEMENT, EN FAISANT
*     ATTENTION AUX DEBORDEMENTS DE TABLEAUX, TRIER PLUS DE
*     [2**(K+1) - 1] ELEMENTS AVEC DES PILES DE CETTE LONGUEUR).

      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM,NINFO),CLE(NCLE),
     +               PTEUR(NELEM)
      DIMENSION TEMP(NCLE), ILOW(16), IUP(16)
      DECLAR MATTRI,TEMP
      DATA NSHELL /10/

*****     VARIABLES DU COMMON     *****

*     CLE : TABLEAU DONT LE IEME INDICE INDIQUE QUEL VECTEUR-CO-
*           LONNE DE 'MATTRI' SERT DE IEME CLE POUR LE TRI
*     MATTRI : MAIRICE CONTENANT LES ELEMENTS (VECTEURS-LIGNES)
*              A TRIER SELON CERTAINS DES INFORMATIONS (VECTEURS-
*              COLONNES)
*     PTEUR : TABLEAU DES POINTEURS DES ELEMENTS A TRIER

*****     VARIABLES LOCALES     *****

*     BINF : BORNE INFERIEURE DES INDICES DES POINTEURS DU SOUS-
*            ENSEMBLE A TRIER
*     BSUP : BORNE SUPERIEURE DES INDICES DES POINTEURS DU SOUS-
*            ENSEMBLE A TRIER
*     DROITE : INDICE DU POINTEUR DE DROITE DONT ON COMPARE L'ELE-
*              MENT CORRESPONDANT A L'ELEMENT ASSOCIE AU POINTEUR
*              DU MILIEU, DANS LE SOUS-ENSEMBLE A TRIER; VAUT 'BSUP'
*              AU DEBUT DU TRI ET PEUT DIMINUER PAR LA SUITE
*     ECHAN : VARIABLE TEMPORAIRE SERVANT AU TRANSFERT DE DEUX POIN-
*             TEURS
*     GAUCHE : INDICE DU POINTEUR DE GAUCHE DONT ON COMPARE L'ELE-
*              MENT CORRESPONDANT A L'ELEMENT ASSOCIE AU POINTEUR
*              DU MILIEU, DANS LE SOUS-ENSEMBLE A TRIER; VAUT 'BINF'
*              AU DEBUT DU TRI ET PEUT AUGMENTER PAR LA SUITE
*     ILOW : VECTEUR SERVANT DE PILE POUR EMMAGASINER L'INDICE DE
*            DEBUT DU SOUS-ENSEMBLE EN ATTENTE D'ETRE TRIE
*     IPILE : INDICE DES VECTEURS SERVANT DE PILE
*     IUP : VECTEUR SERVANT DE PILE POUR EMMAGASINER L'INDICE DE
*           FIN DU SOUS-ENSEMBLE EN ATTENTE D'ETRE TRIE
*     MILIEU : INDICE DU POINTEUR DU MILIEU DU SOUS-ENSEMBLE A
*              TRIER
*     NSHELL : NOMBRE MAXIMUM D'ELEMENTS PERMIS POUR L'UTILISATION
*              DU TRI DE SHELL
*     TEMP : TABLEAU TEMPORAIRE UTLISE POUR LA COMPARAISON DES
*            ELEMENTS SELON LES DIFFERENTES CLES
```

```
*    (SUITE DE LA SOUS-ROUTINE TRIPT)

       DO 6 I=1,NELEM
       PTEUR(I) = I
     6 CONTINUE

       IPILE = 1
       BINF = 1
       BSUP = NELEM

       IF(BINF.GE.BSUP) GOTO 80

*      COMPARAISON DE L'ELEMENT 'BINF' ET DE L'ELEMENT 'MILIEU' ET
*      TRANSFERT DES POINTEURS SI NECESSAIRE

    20 GAUCHE = BINF
       MILIEU = (BINF+BSUP)/2
       CALL AFFECT(MILIEU,TEMP)
       IF(COMPAR(BINF,TEMP)) 30,30,25
    25 CALL TRANSF(BINF,MILIEU)

*      COMPARAISON DE L'ELEMENT 'BSUP' ET DE L'ELEMENT 'MILIEU' ET
*      TRANSFERT DES POINTEURS SI NECESSAIRE

       CALL AFFECT(MILIEU,TEMP)
    30 DROITE = BSUP
       IF(COMPAR(BSUP,TEMP)) 35,50,50
    35 CALL TRANSF(BSUP,MILIEU)

*      SECONDE COMPARAISON DE L'ELEMENT 'BINF' ET DE L'ELEMENT 'MI-
*      LIEU' (CAR CE DERNIER A CHANGE) ET TRANSFERT DES POINTEURS SI
*      NECESSAIRE

       CALL AFFECT(MILIEU,TEMP)
       IF(COMPAR(BINF,TEMP)) 50,50,40
    40 CALL TRANSF(BINF,MILIEU)
       CALL AFFECT(MILIEU,TEMP)
       GOTO 50

    45 CALL TRANSF(DROITE,GAUCHE)

*      TANT QUE L'ELEMENT DE 'DROITE' > CELUI DU 'MILIEU' (SELON LES
*      CLES DEFINIES), DIMINUTION DE 'DROITE' ET COMPARAISON DE CES
*      ELEMENTS

    50 DROITE = DROITE - 1
       IF(COMPAR(DROITE,TEMP)) 60,60,50

*      TANT QUE L'ELEMENT DE 'GAUCHE' < CELUI DU 'MILIEU' (SELON LES
*      CLES DEFINIES), AUGMENTATION DE 'GAUCHE' ET COMPARAISON DE CES
*      ELEMENTS

    60 GAUCHE = GAUCHE + 1
       IF(COMPAR(GAUCHE,TEMP)) 60,65,65

*      COMPARAISON DES INDICES 'GAUCHE' ET 'DROITE' : LORSQUE CEUX-
*      CI SE RENCONTRENT, LES ELEMENTS ONT TOUS ETE EXAMINES ET ON
*      SEPARE LE SOUS-ENSEMBLE EN DEUX

    65 IF(GAUCHE.LE.DROITE) GOTO 45
       IF(DROITE-BINF.LE.BSUP-GAUCHE) GOTO 70
       ILOW(IPILE) = BINF
       IUP(IPILE) = DROITE
       BINF = GAUCHE
       IPILE = IPILE + 1
       GOTO 90
```

```
*    (SUITE DE LA SOUS-ROUTINE TRIPT)

     70 ILOW(IPILE) = GAUCHE
        IUP(IPILE) = BSUP
        BSUP = DROITE
        IPILE = IPILE + 1
        GOTO 90

*    FIN DU TRI D'UN SOUS-ENSEMBLE : ON DEPILE POUR VOIR S'IL
*    RESTE D'AUTRES SOUS-ENSEMBLES A TRIER

     80 IPILE = IPILE - 1
        IF(IPILE.EQ.0) RETURN
        BINF = ILOW(IPILE)
        BSUP = IUP(IPILE)

*    TRI DU SOUS-ENSEMBLE LE PLUS PETIT FORME DE LA DIVISION DE
*    L'ANCIEN SOUS-ENSEMBLE EN DEUX; TRI DE SHELL DE TYPE T.C.
*    SI LE NOMBRE D'ELEMENTS DU SOUS-ENSEMBLE <= 10

     90 IF(BSUP-BINF.GE.NSHELL) GOTO 20
        BINF = BINF - 1

    100 BINF = BINF + 1
        IF(BINF.EQ.BSUP) GOTO 80
        CALL AFFECT(BINF+1,TEMP)
        IF(COMPAR(BINF,TEMP)) 100,100,105

    105 GAUCHE = BINF
        ECHAN = PTEUR(GAUCHE+1)

    110 PTEUR(GAUCHE+1) = PTEUR(GAUCHE)
        GAUCHE = GAUCHE - 1
        IF(GAUCHE.EQ.0) GOTO 120
        IF(COMPAR(GAUCHE,TEMP)) 120,120,110

    120 PTEUR(GAUCHE+1) = ECHAN

        GOTO 100

        END


        SUBROUTINE AFFECT(IAFFEC,TEMP)

*    CETTE SOUS-ROUTINE AFFECTE LES INFORMATIONS SERVANT DE
*    CLES, POUR L'ELEMENT DE LA MATRICE A TRIER POINTE PAR
*    'IAFFEC', AU TABLEAU TEMPORAIRE 'TEMP'

        IMPLICIT INTEGER (A-Z)
        COMMON /PARAM/ MATTRI(NELEM,NINFO),CLE(NCLE),
       +               PTEUR(NELEM)
        DIMENSION TEMP(NCLE)
        DECLAR MATTRI,TEMP

****    VARIABLES LOCALES    ****

*    IAFFEC : INDICE DU POINTEUR POUR LEQUEL ON VEUT COMPARER L'E-
*             LEMENT CORRESPONDANT AVEC UN AUTRE
*    KLE : VARIABLE TEMPORAIRE CONTENANT LA IEME CLE

        DO 10 I=1,NCLE
        KLE = CLE(I)
        TEMP(I) = MATTRI(PTEUR(IAFFEC),KLE)
     10 CONTINUE

        RETURN
        END
```

```
      INTEGER FUNCTION COMPAR(ICOMP,TEMP)
*     CETTE FONCTION COMPARE DEUX ELEMENTS SELON LES CLES DEJA
*    DEFINIES
      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM,NINFO),CLE(NCLE),
     +               PTEUR(NELEM)
      DIMENSION TEMP(NCLE)
      DECLAR MATTRI,TEMP
****     VARIABLES LOCALES     ****
*     ICOMP :  INDICE DU POINTEUR POUR LEQUEL L'ELEMENT CORRESPON-
*              DANT EST COMPARE, SELON CHAQUE CLE, AU TABLEAU TEMPO-
*              RAIRE 'TEMP'
*     KLE : VARIABLE TEMPORAIRE CONTENANT LA IEME CLE
      DO 10 I=1,NCLE
      KLE = CLE(I)
      IF(MATTRI(PTEUR(ICOMP),KLE)-TEMP(I)) 20,10,30
   10 CONTINUE
      COMPAR = 0
      RETURN
   20 COMPAR = -1
      RETURN
   30 COMPAR = 1
      RETURN
      END

      SUBROUTINE TRANSF(ITRAN1,ITRAN2)
*     CETTE SOUS-ROUTINE FAIT LE TRANFERT DES DEUX POINTEURS
*    DONT LES INDICES SONT 'ITRAN1' ET 'ITRAN2'
      IMPLICIT INTEGER (A-Z)
      COMMON /PARAM/ MATTRI(NELEM,NINFO),CLE(NCLE),
     +               PTEUR(NELEM)
      DECLAR MATTRI
****     VARIABLES LOCALES     ****
*     BIDON : VARIABLE TEMPORAIRE SERVANT AU TRANSFERT
*     ITRAN1,ITRAN2 :  INDICES DES POINTEURS POUR LESQUELS ON VEUT
*                      EFFECTUER UN TRANSFERT
      BIDON = PTEUR(ITRAN1)
      PTEUR(ITRAN1) = PTEUR(ITRAN2)
      PTEUR(ITRAN2) = BIDON
      RETURN
      END
```

BIBLIOGRAPHIE

KNUTH, D., Sorting and Searching, Vol. 3, pp. 114-123, Addison Wesley (1973).

WIRTH, N., Algorithms + Data Structures = Programs, pp. 79-82, Prentice-Hall (1976).

- - - - - - - - - - - - - - -